我们创建了一个名为 canvas 的 SFML VertexArray 实例。我们称这个 VertexArray 为 canvas，因为它实际上将是整个游戏的画布。每帧游戏中，所有游戏对象都将被添加到 VertexArray 中，然后使用 canvas 将其绘制到窗口上。接下来，我们声明了一个即将编写的 InputDispatcher 类的实例。我们很快会在主游戏循环中看到它是如何被使用的。现在，只需注意我们将 RenderWindow 的地址传递给了它的构造函数。然后，我们声明了一个 GameObject 实例的向量。如前所述，我们游戏中的每个实体都将包含在一个 GameObject 实例中。

inputDispatcher 实例调用了 dispatchInputEvents 函数。在我们即将编写的这个函数内部，所有输入事件都会与之前声明过感兴趣的任何游戏对象共享。Factory 类负责让游戏对象与 inputDispatcher 建立连接，然后每个游戏对象处理它关心的输入。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 电子邮件

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

文本, 信件

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。GameObject 中，该对象在游戏循环的每次迭代中运行一次包含3个，添加事件，更新游戏逻辑（根据上一帧的时间）。画出相应的组件。

文本

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

文本, 信件

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本

AI 生成的内容可能不正确。

图形用户界面, 文本

AI 生成的内容可能不正确。

文本, 信件

AI 生成的内容可能不正确。